Angular Fundamentals
Module 1 - Inleiding

Peter Kassenaar
info@kassenaar.com

Peter Kassenaar

- Trainer, author, developer – since 1996
- Specialty: *"Everything JavaScript"*
- JavaScript, ES6, Angular, NodeJS, TypeScript, React, Vue, Phonegap

www.kassenaar.com

info@kassenaar.com

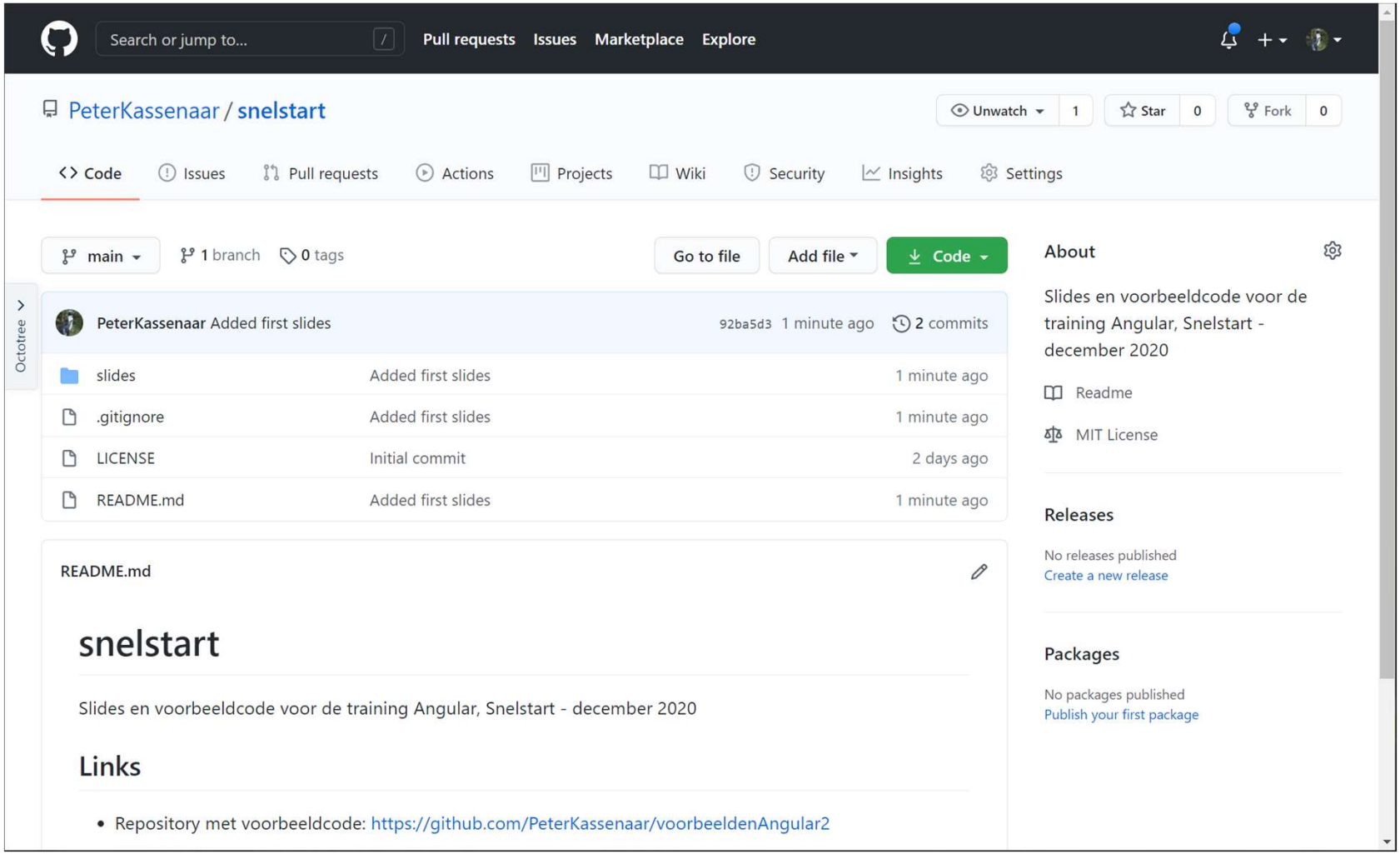
Twitter: @PeterKassenaar

ING OHRA euricom A DIMENSION DATA COMPANY sanoma delta lloyd zenito BETERE ZEKERHEID VOOR ONDERNEMERS Atos woonbron OBERON INTERACTIVE ROC West-Brabant the eforum FACTORY

https://didacticum.nl

www.angulartraining.nl

github.com/PeterKassenaar/snelstart



Over jullie

Stel jezelf kort voor

Voorkennis webdevelopment, (mobile/web-) apps?

(Kennis AngularJS 1.x?)

Voorkennis andere (web)talen?

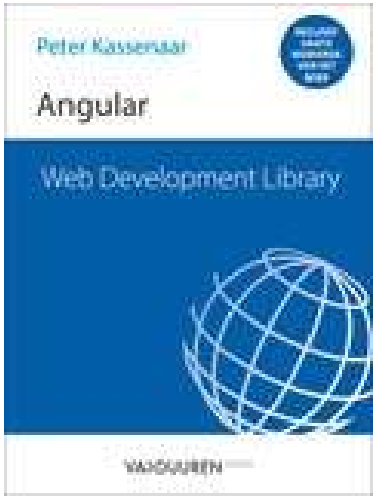
Verwachtingen van de cursus?

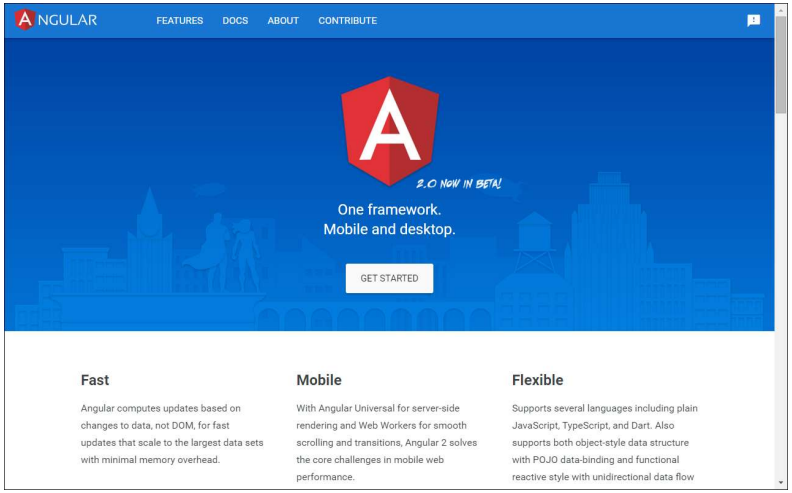
Concrete projecten?

Specifieke vragen of technieken die je wilt behandelen?

Materialen

Software	(Angular, NodeJS + NPM, editor, browser)
Handouts	(Github - PDF)
Oefeningen	(Github)
Websites	(online)





angular.io/

Agenda - globally

30 November – 2 december – ma - wo

"*Fundamentals*"

December 2020 + januari 2021 - ma-wo

"*Advanced*"

~9:00 start

~ 10:30 Break

~12:00 lunch

~ 14:30 Break

~16:15-16:30 einde

Doel van de training

*Je wordt **geen** Angular wizard in 3 dagen (sorry)*

maar….

Goals

1. Je leert over de structuur en architectuur van Angular Apps. Van een kleine `hello-world` app tot een grote Enterprise applicatie.

2. Je bent bekend met de belangrijkste Angular concepten van het framework. Specifieke details kun je altijd Googelen.

3. Je hebt enige hands-on ervaring met het maken van apps en componenten, services, API's/backends, security concepten, routing en formulieren.

4. Je hebt een algemeen begrip van de manier waarop moderne web apps worden gemaakt met Angular, TypeScript en build tools.

Agenda - 3 dagen

- Introductie & geschiedenis - waarom Angular?
- Kernbegrippen in Angular 2 – 10
- CLI, Hello World in Angular – inzicht in boilerplate-code
- Angular in depth (modueles:
  - Components
  - ECMAScript 2015 + TypeScript
  - Data binding
  - Dependency Injection (DI) – more components
  - Services en http, Observables (RxJS), communicatie met backend
  - Routing, [Reactive] Forms
- BEST PRACTICES / STYLE GUIDE

Agenda

- Day 1 - Intro
  - Theory - Introductie & geschiedenis - why Angular
  - Hello World in Angular –boilerplate-code
  - Concepts, context & architecture
  - Angular CLI
  - Components
  - Data binding

Agenda

- Day 2 – Services & Communication
  - Services & DI
  - Observables (RxJS)
  - HTTP, Live API’s,
  - Applications as a tree of components

Agenda

- Day 3 – Next Steps topics
  - Intercomponent communication
    - Creating event buses
  - Routing
  - (Forms
    - Template Driven Forms
    - Model Driven Forms)

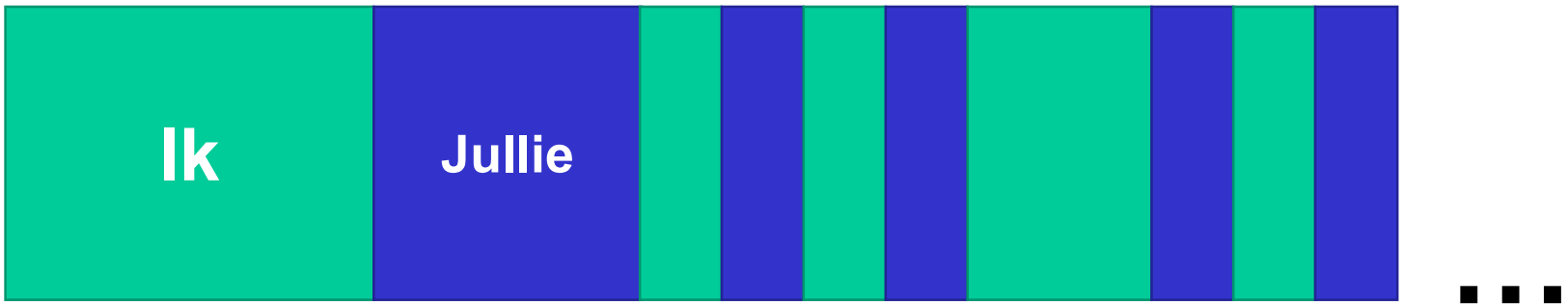
- Later in dec. + Jan. – Angular Advanced topics

2 Richtlijnen

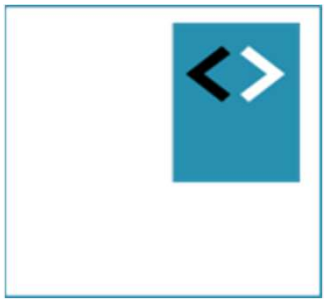
1. Oefeningen
    - Maar: neem ook vooral zijpaden, experimenteer, lees verder, maak een eigen project, app, website…

2. Voorbeeldcode
    - Als ondersteuning bij de oefeningen, zie boven
    - Work in progress – check de Angular-site!
    - github.com/PeterKassenaar/voorbeeldenAngular2

Globale werkwijze



Vragen?



Angular vs. The Rest

Differences, similarities, new features

Addressing the "WHY" question!

WHY, would we want to use a frontend framework.

It is all HTML, CSS and JavaScript right?

Rethorical question:

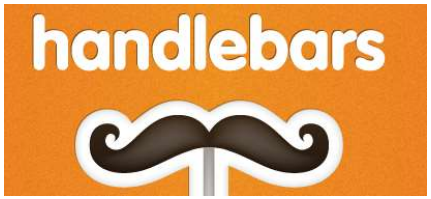
"Do we want to go back

to the jQuery days?"

speed, consistency, not re-inventing the wheel, community, performance, testing….

Old school web apps

HTML + templates



Data Binding

Routing

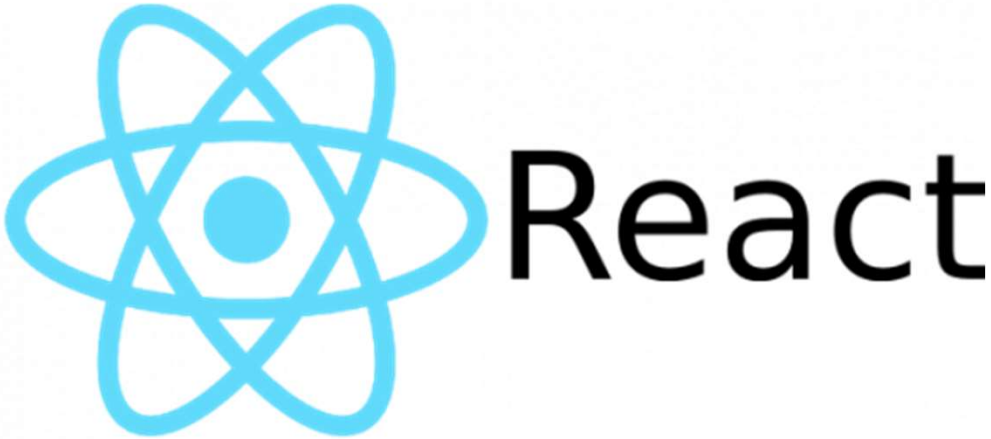
DOM-manipulation

Mobile development

...

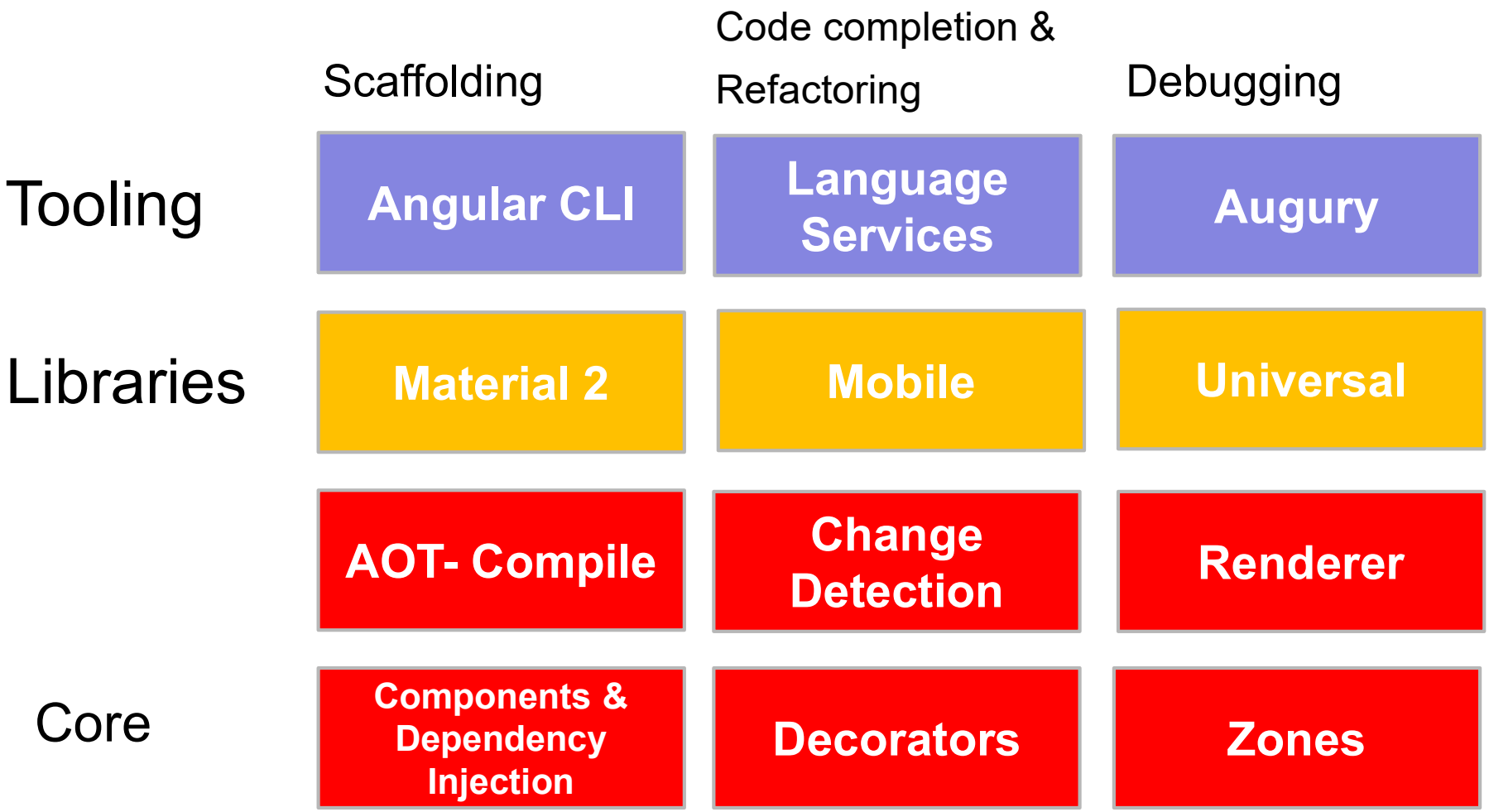
“The Frankenstein Framework”

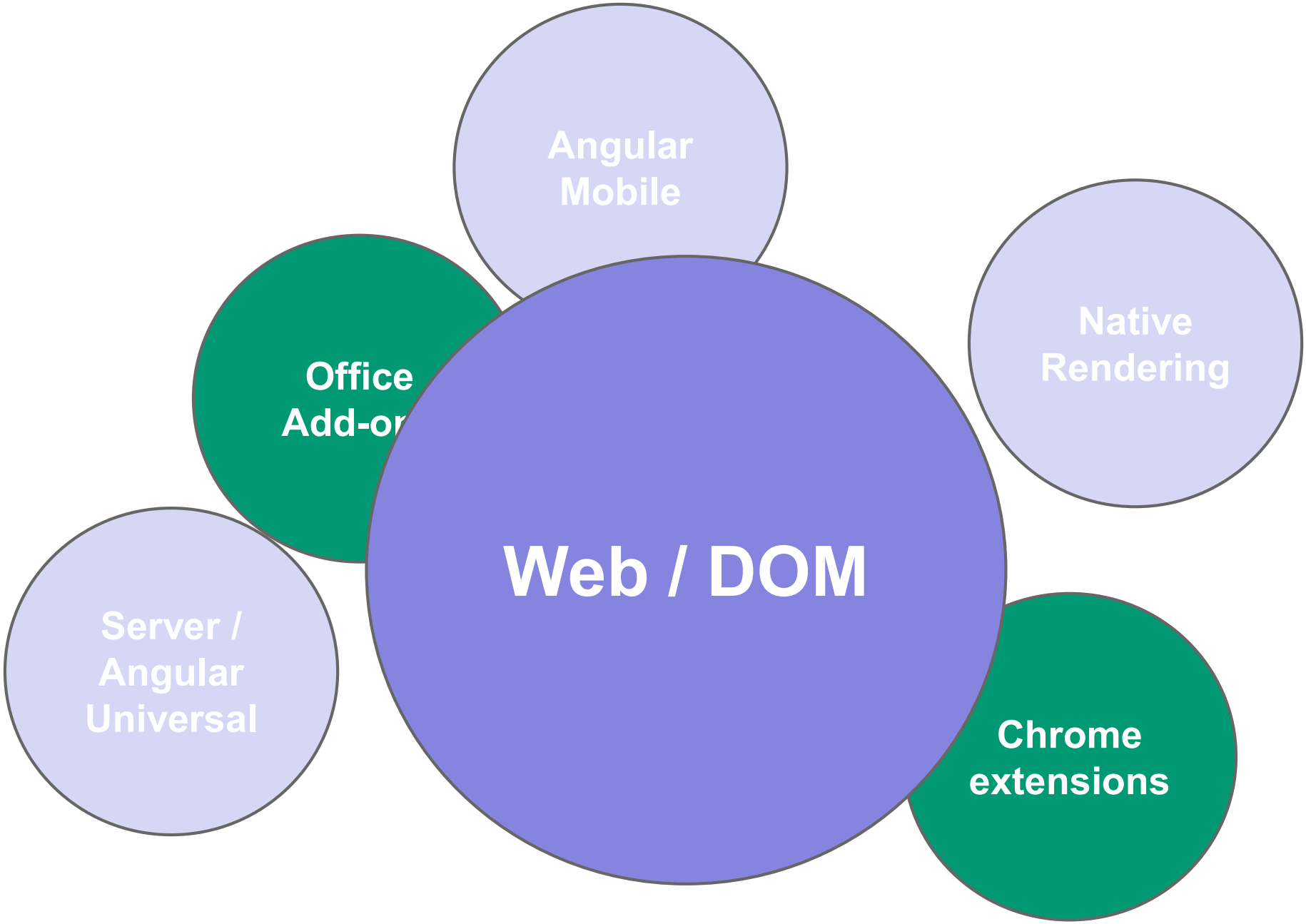
Front-end Frameworks – the big four



Platform

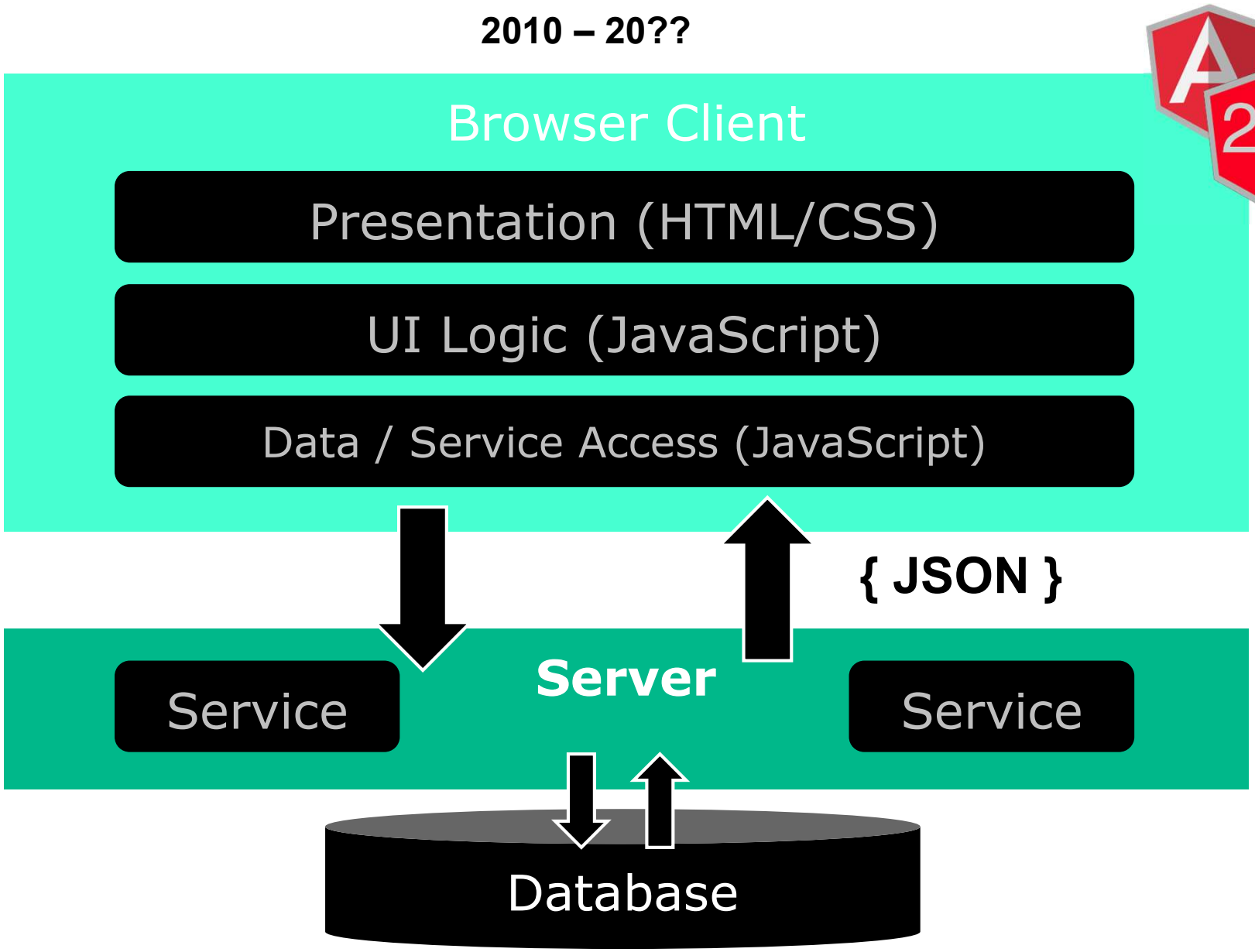
Platform Features



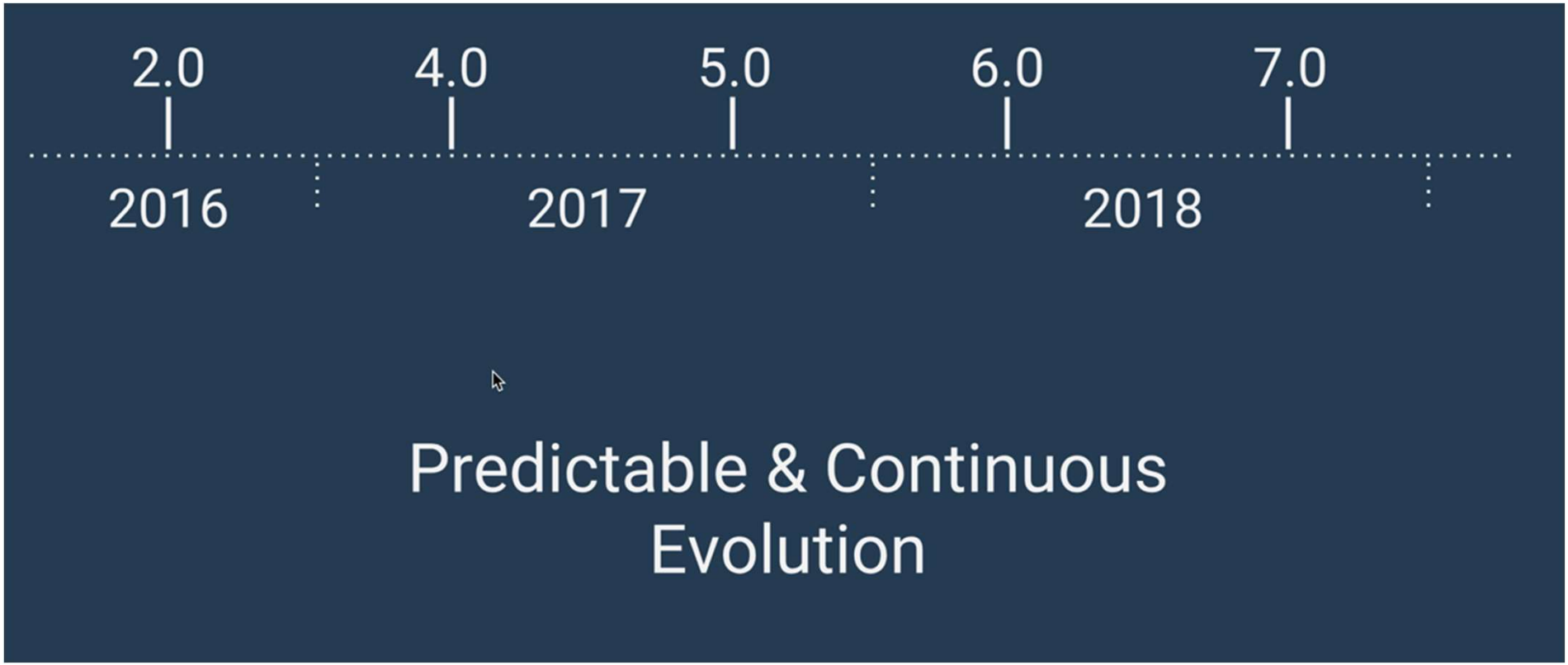

Angular
Mobile
Native
Rendering
Office
Add-or
Web / DOM
Server /
Angular
Universal
Chrome
extensions

Single Page Application

2010 – 20??



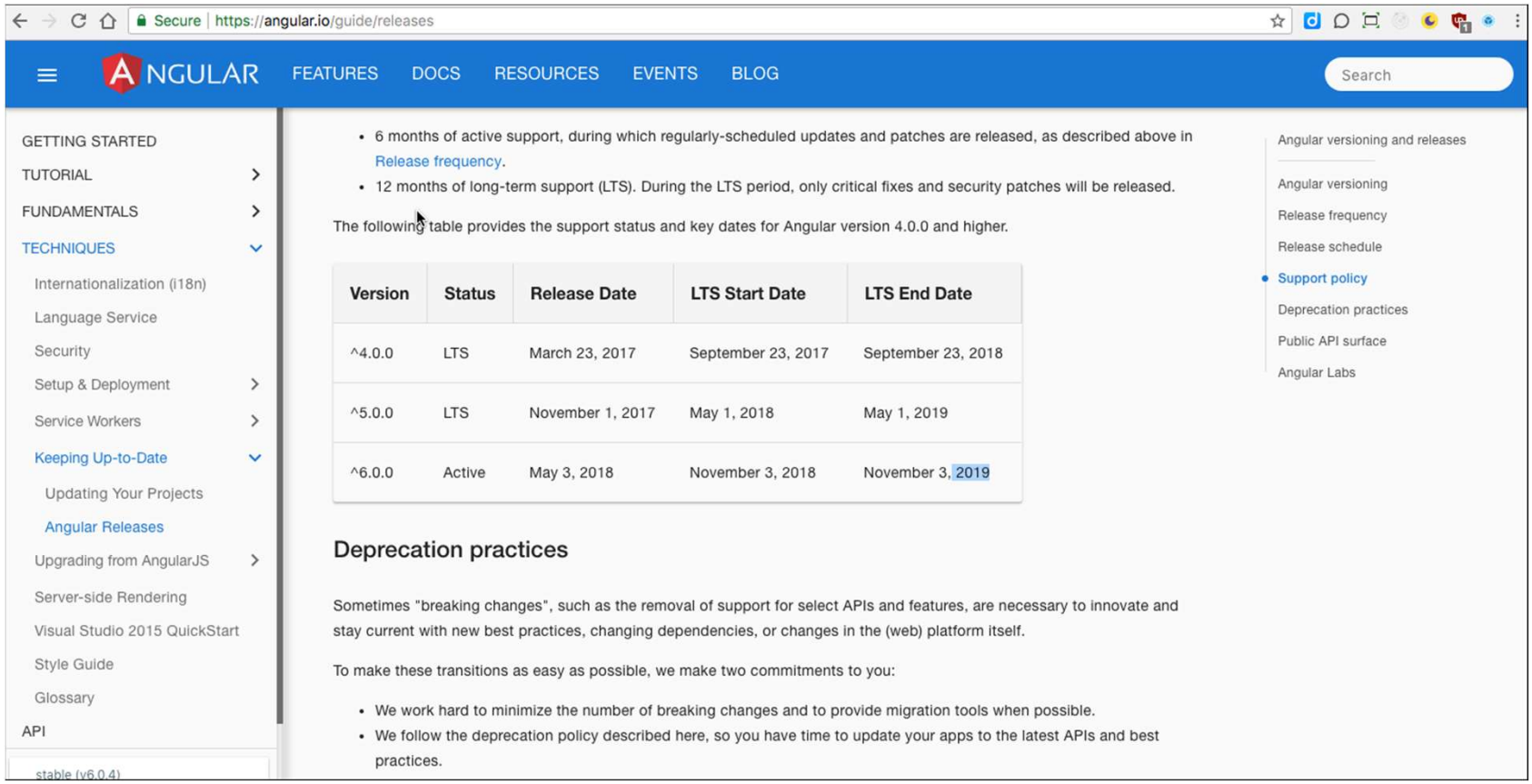
@IgorMinar, Nov. 9, 2018


2.0
4.0
5.0
6.0
7.0
2016
2017
2018
Predictable & Continuous
Evolution

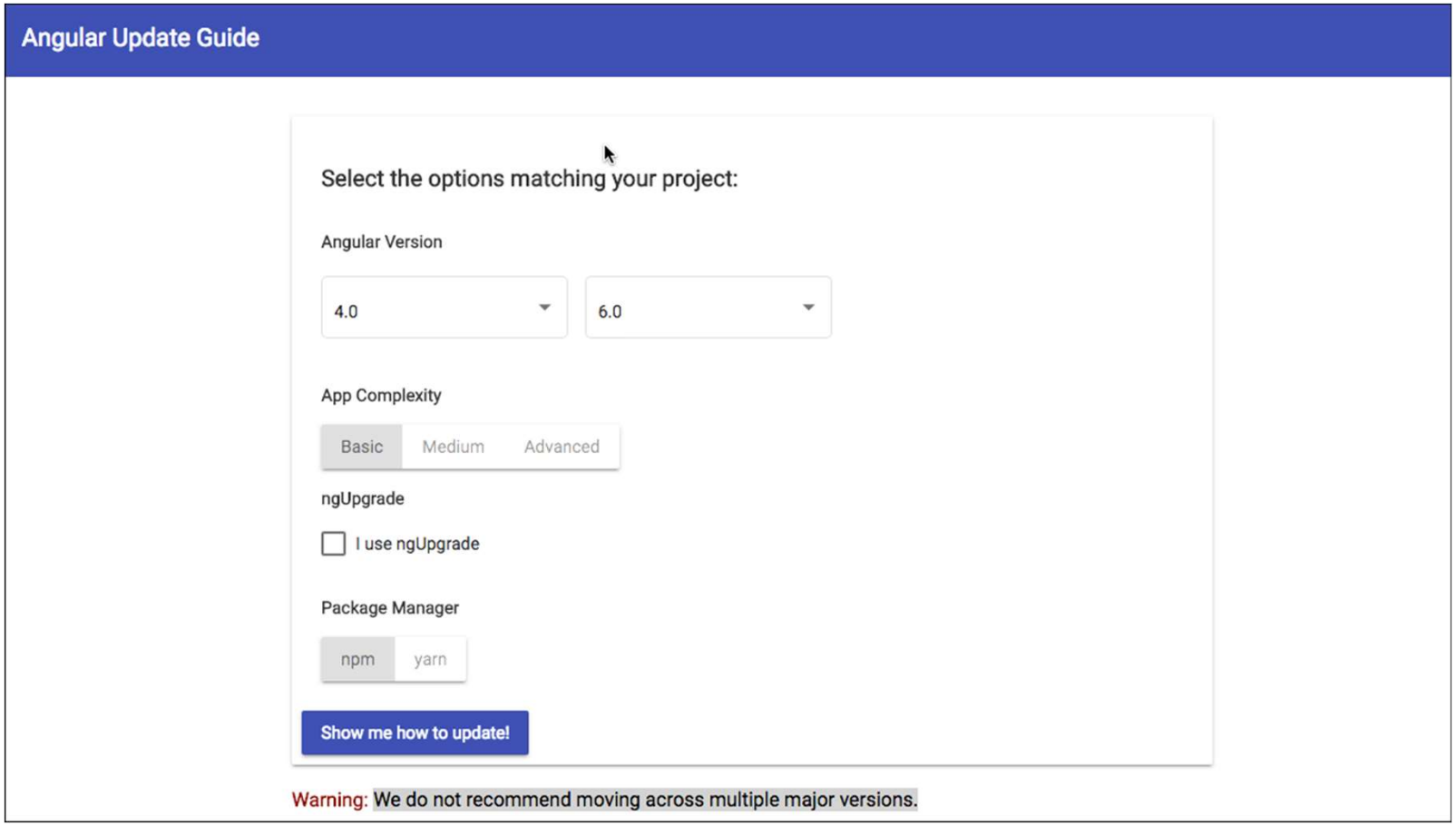

2.0
4.0
5.0
6.0
7.0
2016
2017
2018
✅ Predictability
✅ Painless Updates
✅ Long Term Support

Angular Versies en -Long Time Support

→ https://angular.io/guide/releases



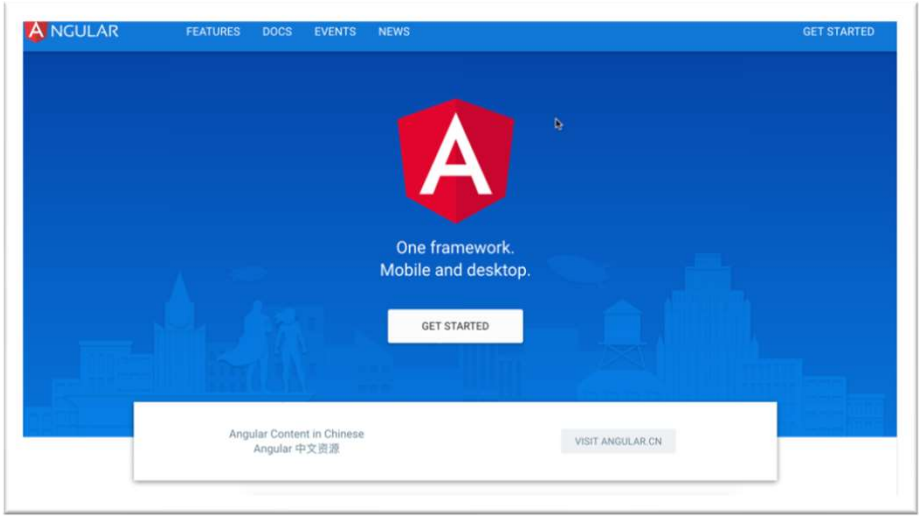
https://update.angular.io/



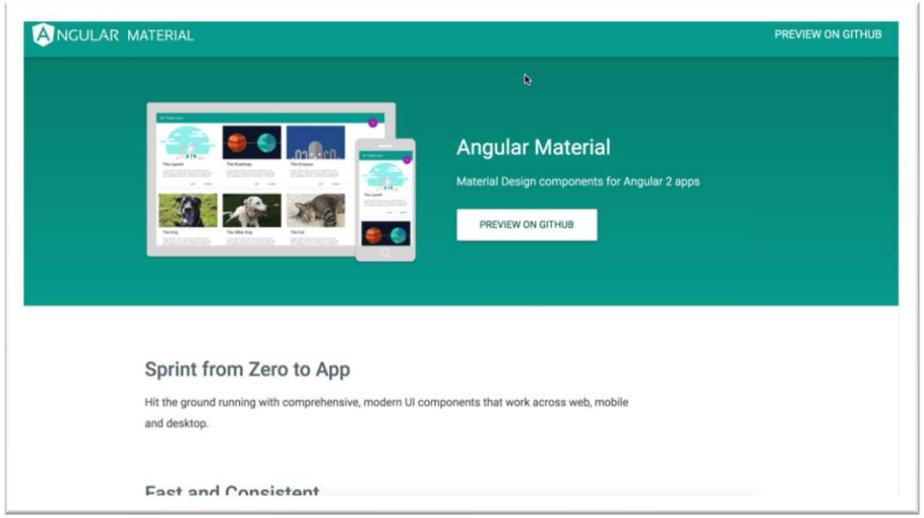
"It's just

Angular

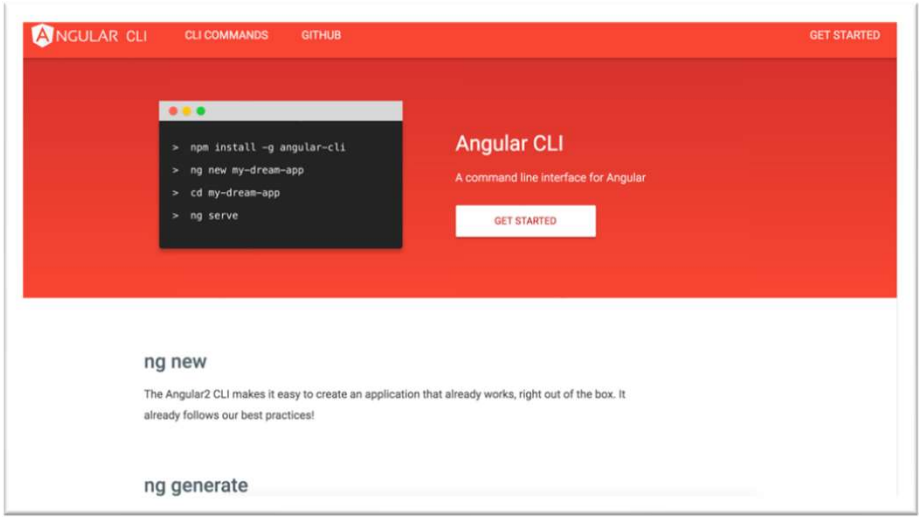
Angular as a Platform



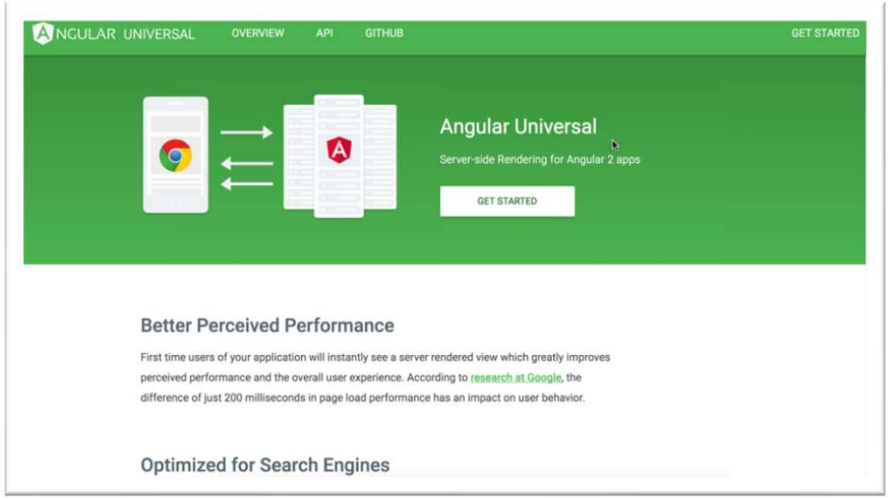
https://angular.io/



https://material.angular.io/

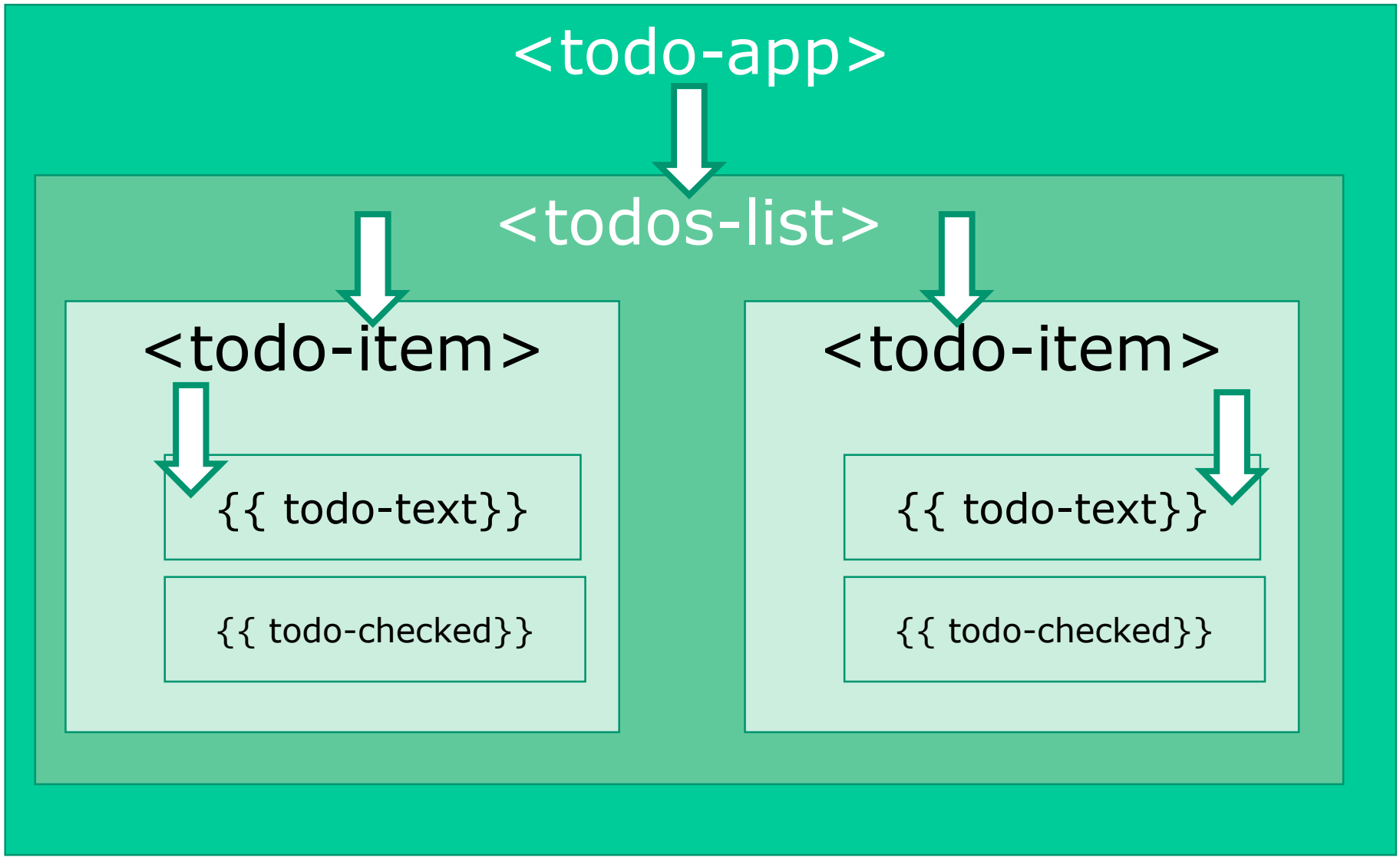


https://cli.angular.io/



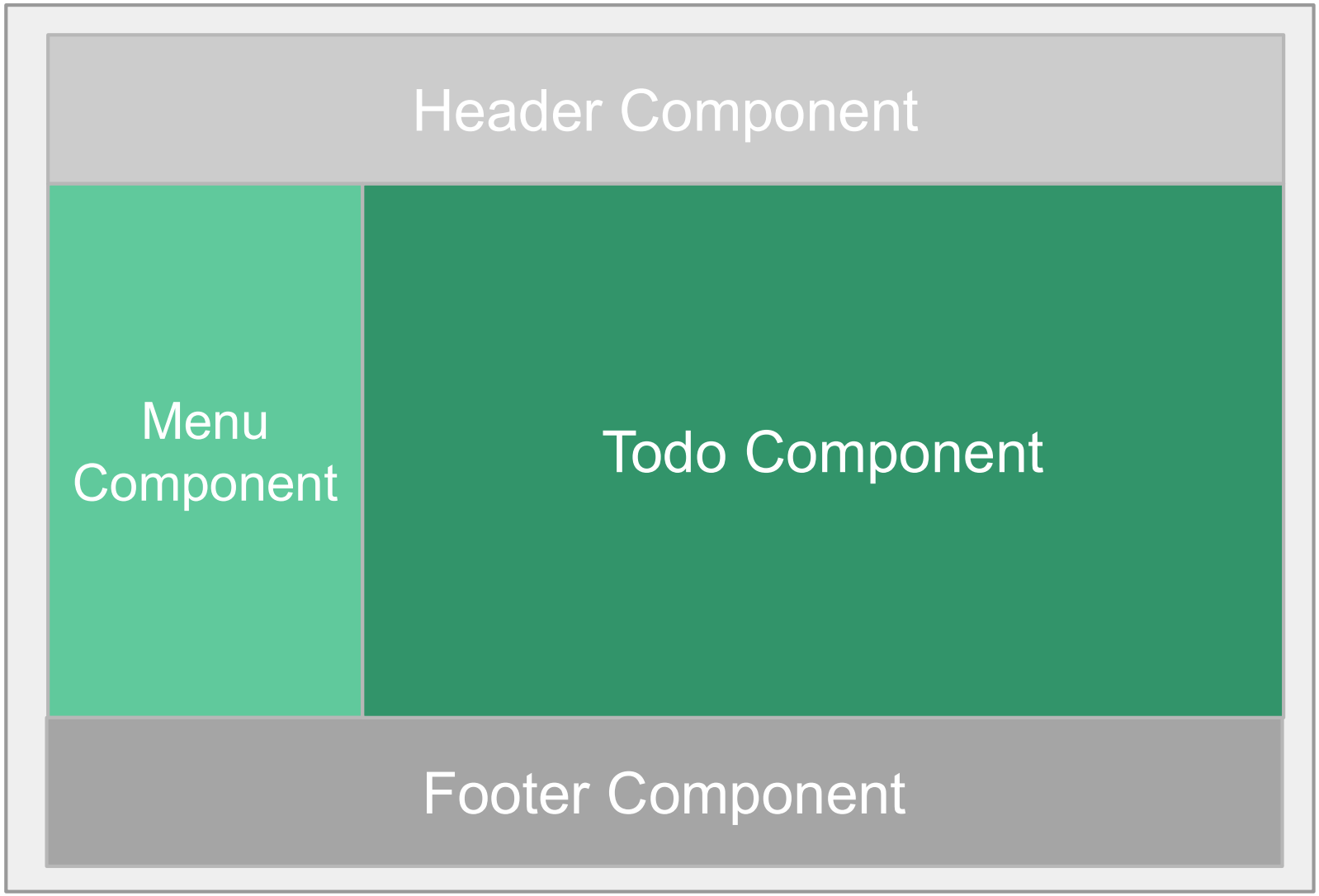
https://universal.angular.io/

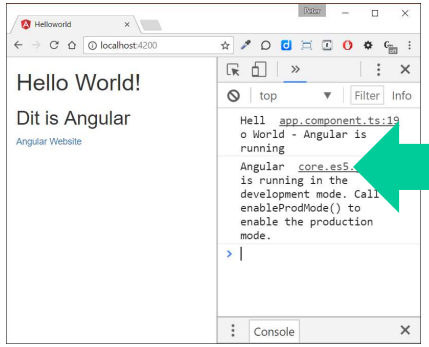
Angular 2 - components



"An Angular-app is a tree of components"

Components – visual representation





main.ts / bootstrapper

ngModule / root module

AppComponent

Other components

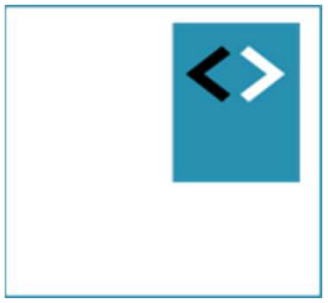
Other components

Other components

Services

3rd party libs

Other modules



Let’s write some code

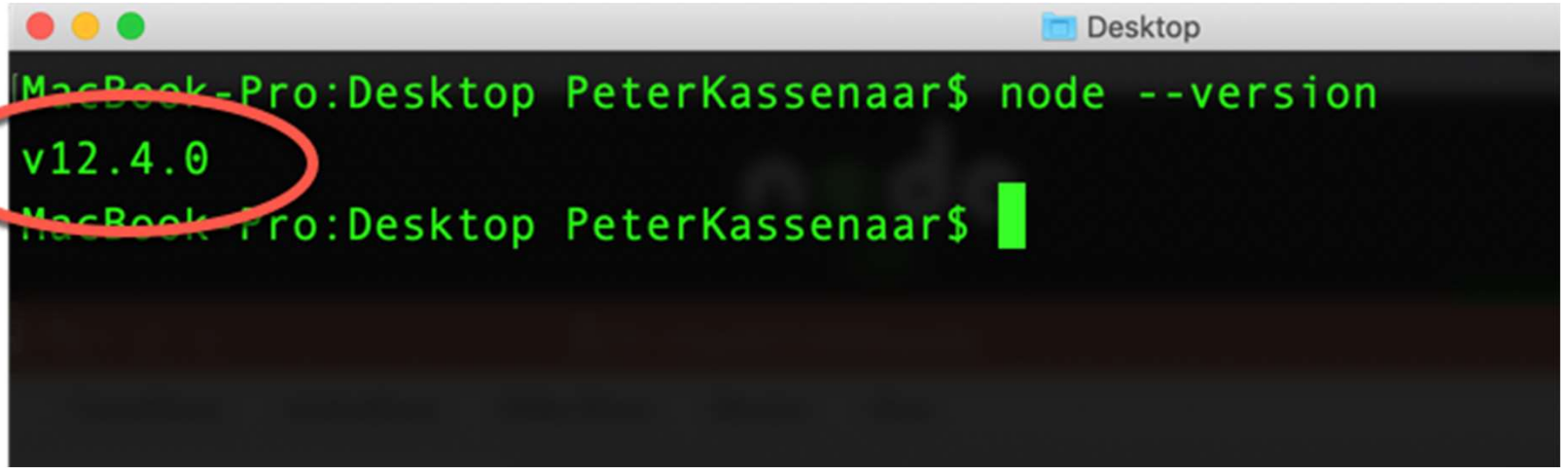
Hello World in Angular

Angular 1:

```
<script  src="angular.min.js></script>
```

Angular development dependency: NodeJS 10+

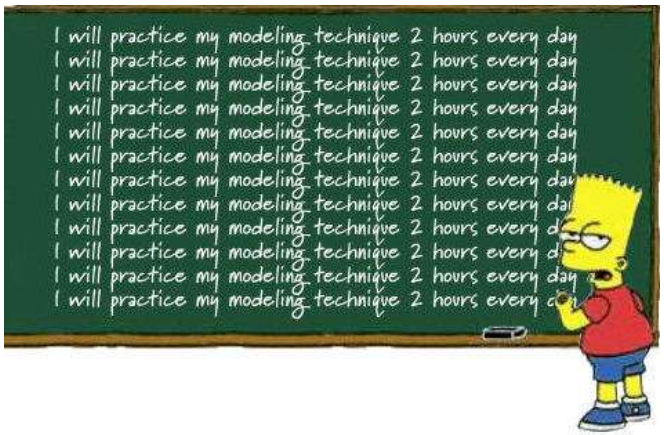
Node – check your version



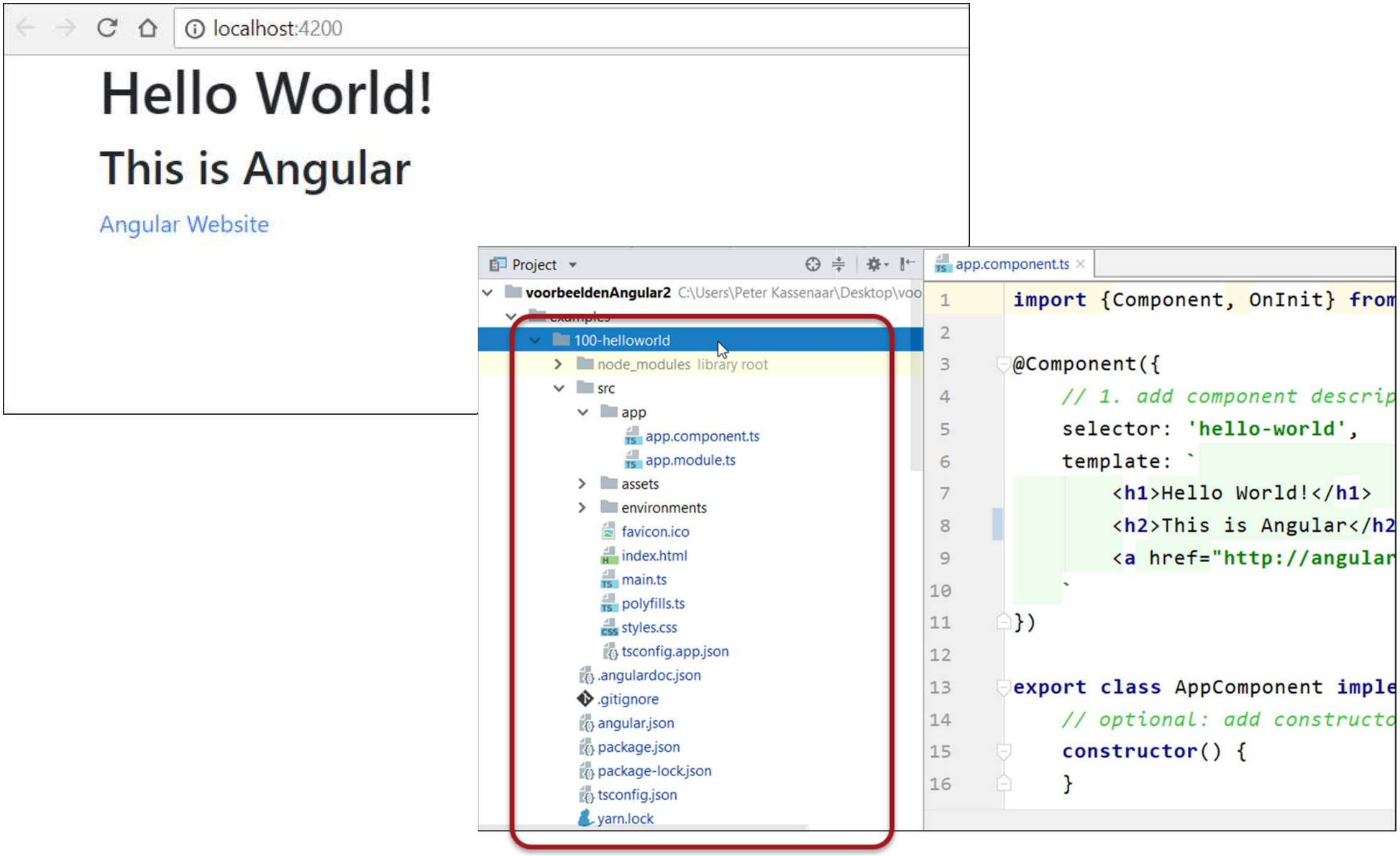
Mini workshop

- Download or clone https://github.com/PeterKassenaar/voorbeeldenAngular2

```
cd examples

cd 100-helloworld

npm install

npm start
```



- Go to browser: http://localhost:4200


localhost:4200
Hello World!
This is Angular
Angular Website
Project
voorbeeldenAngular2 C:\Users\Peter Kassenaar\Desktop\voo
100-helloworld
node_modules library root
src
app
app.component.ts
app.module.ts
assets
environments
favicon.ico
index.html
main.ts
polyfills.ts
styles.css
tsconfig.app.json
.angulardoc.json
.gitignore
angular.json
package.json
package-lock.json
tsconfig.json
yarn.lock
app.component.ts
import {Component, OnInit} from
@Component({
// 1. add component descrip
selector: 'hello-world',
template: `
<h1>Hello World!</h1>
<h2>This is Angular</h2
<a href="http://angular
`
})
export class AppComponent imple
// optional: add constructo
constructor() {
}

Boilerplate code for Hello World

Steps

1. Set up environment, boilerplate & libraries
    - Important configuration files
2. Angular Component(s)
3. Angular Module(s): @ngModule()
4. Bootstrap our module
5. Write HTML-pagina (`index.html`)

Boilerplate files #1 - `package.json`

```
{
  "name": "hello-angular",
  "description": "Voorbeeldproject bij de training Angular (C) - info@kassenaar.com",
  "version": "0.0.1",
  "license": "MIT",
  "scripts": {
    "ng": "ng",
    "start": "ng serve",
    "build": "ng build",
  },
  "private": true,
  "dependencies": {
    "@angular/animations": "6.0.0",
    "@angular/common": "6.0.0",
    "@angular/compiler": "6.0.0",
    "@angular/core": "6.0.0",
    "@angular/forms": "6.0.0",
    "rxjs": "^6.1.0",
    "zone.js": "^0.8.26"
  },
  "devDependencies": {
    "@angular-devkit/build-angular": "~0.6.0",
    "@angular/cli": "6.0.0",
    "typescript": "2.7.2"
  },
  "author": "Peter Kassenaar <info@kassenaar.com>"
}
```

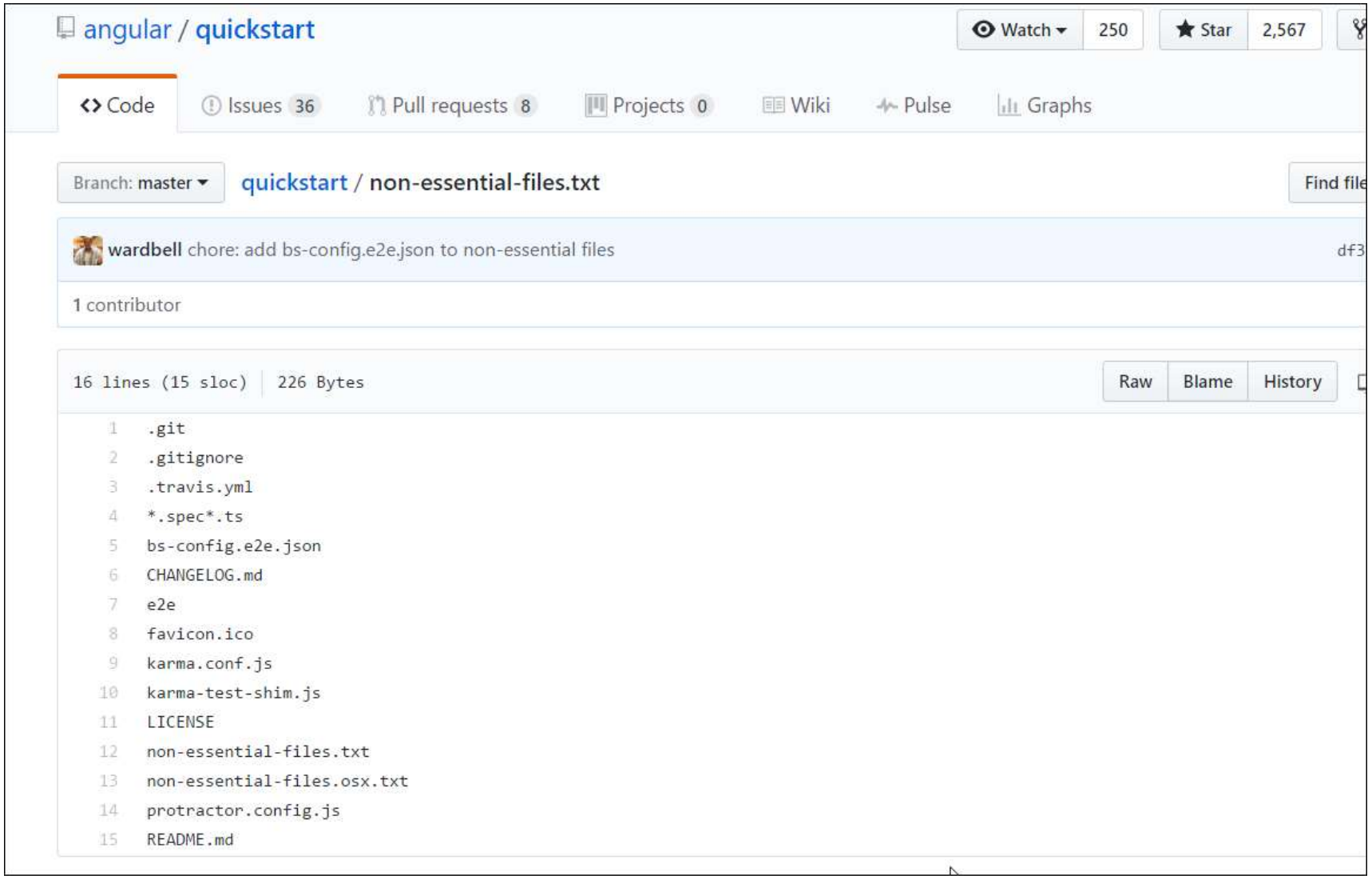
Boilerplate files #2 - **tsconfig.json**

```
{
  "compileOnSave"  : false,
  "compilerOptions": {
   "outDir"                 : "./dist/out-tsc",
   "baseUrl"                : "src",
   "sourceMap"              : true,
   "declaration"            : false,
   "moduleResolution"       : "node",
   "emitDecoratorMetadata" : true,
   "experimentalDecorators": true,
   "target"                 : "es5",
   "typeRoots"              : [
     "node_modules/@types"
   ],
   "lib"                    : [
     "es2016",
     "dom"
   ]
  }
}
```

Boilerplate files #3 – **angular.json**

```
{
  "$schema": "./node_modules/@angular/cli/lib/config/schema.json",
  "version": 1,
  "newProjectRoot": "projects",
  "projects": {
    "helloworld": {
      "root": "",
      "sourceRoot": "src",
      "projectType": "application",
      "architect": {
        "build": {
          "builder": "@angular-devkit/build-angular:browser",
          "options": {
            "outputPath": "dist",
            "index": "src/index.html",
            "main": "src/main.ts",
            "tsConfig": "src/tsconfig.app.json",
      …
}
```

"Nice to have" - non-essential files



https://github.com/angular/quickstart/blob/master/non-essential-files.txt

Step 2 – Component

Convention - components in directory `/src/app`

Or: edit in `angular.json`

Filename: `src/app/app.component.ts`

```
import {Component} from ‘@angular/core';
@Component({
   selector: 'hello-world',
   template: '<h1>Hello Angular</h1>'
})
export class AppComponent {

}
```

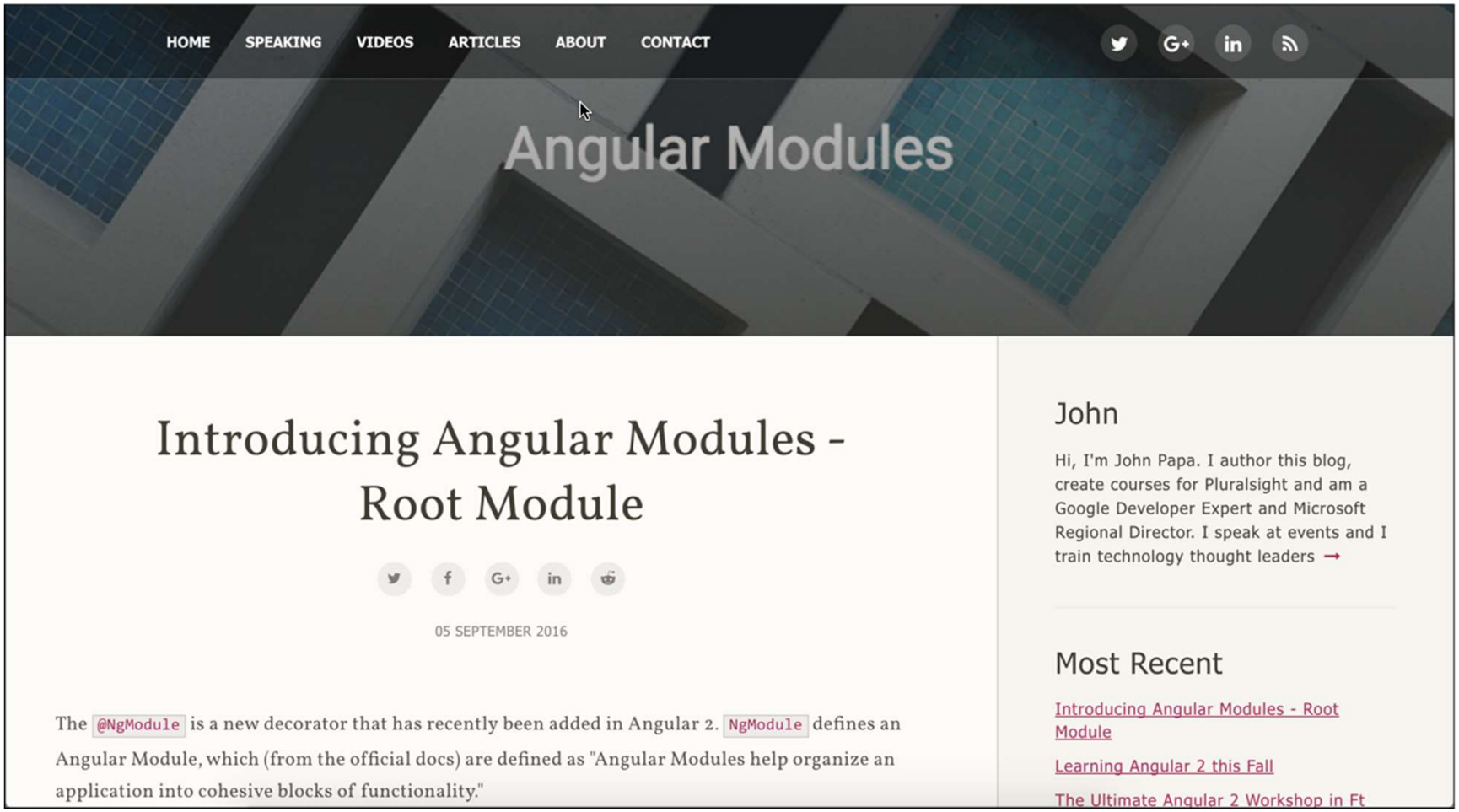
Step 3 – @ngModule

Convention - filename: `/src/app.module.ts`

```
// Angular Modules
import {NgModule}      from '@angular/core';
import {BrowserModule} from '@angular/platform-browser';

// Custom Components
import {AppComponent} from './app.component';

// Module declaration
@NgModule({
   imports     : [BrowserModule],
   declarations: [AppComponent],
   bootstrap   : [AppComponent]
})
export class AppModule {
}
```

Root Module of the application

Some background info on Root Module



https://johnpapa.net/introducing-angular-modules-root-module/

Step 4 - bootstrap component

Best practice: bootstrap app in separate component

Convention: `main.ts`, of `app.main.ts`.

```
import {enableProdMode} from '@angular/core';
import {platformBrowserDynamic} from '@angular/platform-browser-dynamic';

import {AppModule} from './app/app.module';
import {environment} from './environments/environment';

if (environment.production) {
   enableProdMode();
}

platformBrowserDynamic().bootstrapModule(AppModule);
```

Step 5 – index.html

`index.html` - simple HTML file - expanded at runtime by WebPack

Header:

```
<html>

<head>
  <meta charset="utf-8">
  <title>Helloworld</title>
  <base href="/">

  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
```

Body van index.html

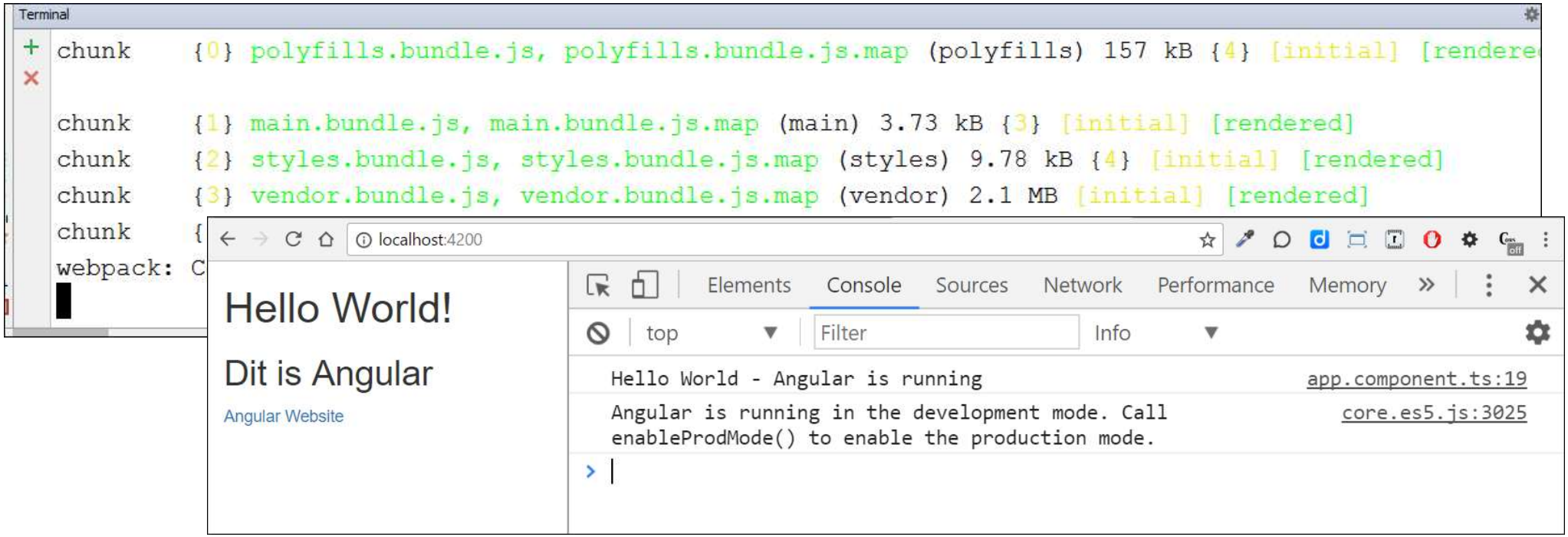
Verwijzing naar de root-component:

```
<body>
  <hello-world>
    Bezig met laden...
  </hello-world>
</body>
```

App draaien

`npm start` – draait de scriptopdracht start uit `package.json`.

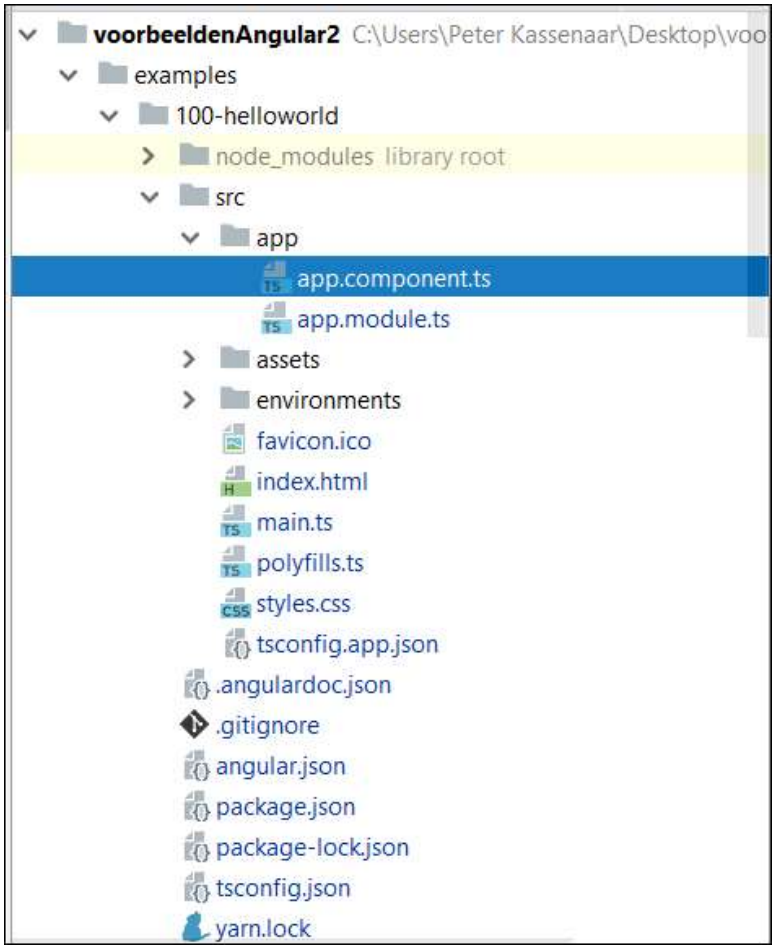
`ng serve` - start globale angular-cli instantie

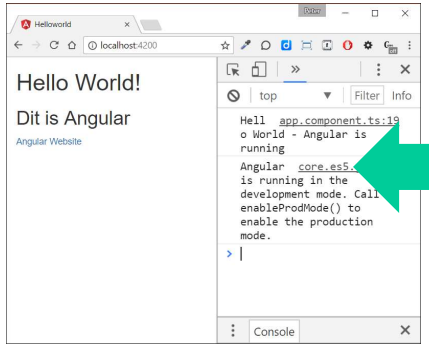


Daarna: wijzigingen aanbrengen in `app.component.ts`

– worden opgepikt door Live Reload

Basic Project Structure





main.ts / bootstrapper

ngModule / root module

AppComponent

Services

Components

Other modules

Other components

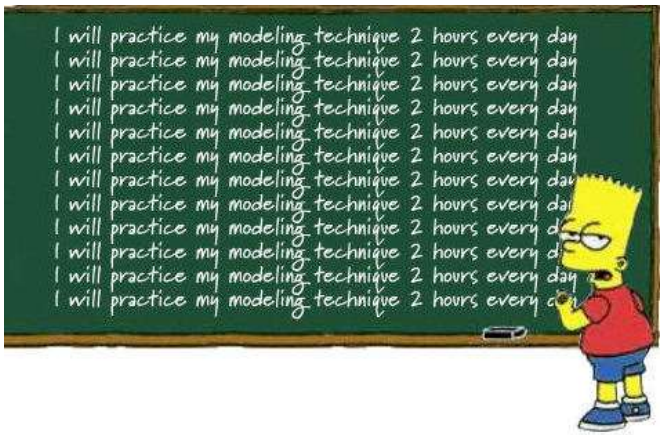
Other components

Other components

Checkpoint

- Er is aardig wat boilerplate code nodig om een Angular-app te starten
- Vier stappen
    1. Set up environment, boilerplate & libraries
    2. Schrijf Angular Root Component voor de app
    3. Bootstrap de component
    4. Schrijf HTML-pagina (`index.html`)
- Daarna: app gaan uitbreiden
- Oefening `1a), 1b), 1c), 1d)`

Oefening….



Assets

github.com/PeterKassenaar/voorbeeldenAngular2

Oefeningen en meer voorbeeldcode

Angular CLI

Snel nieuwe projecten instellen via de command line

Angular-CLI to the rescue

- Het *is* mogelijk nieuwe Angular-projecten from scratch te starten.
- Met de CLI is eenvoudiger.
- CLI-options:
  - Scaffolding
  - Generating
  - Testing
  - Building
  - AOT-Compiling
  - …

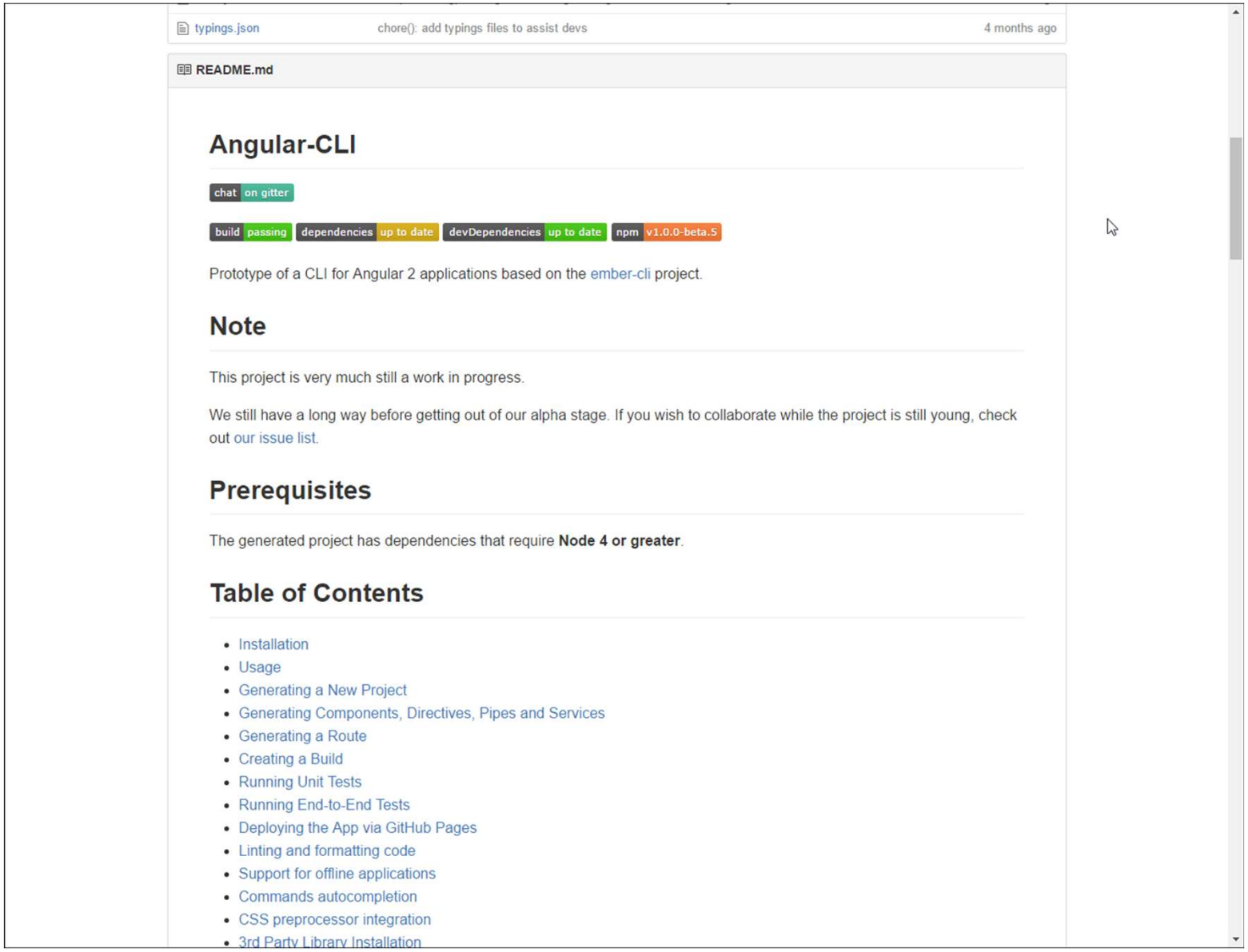
Scaffolding - Angular CLI

Projecten, componenten, routes en meer definiëren vanaf de command line

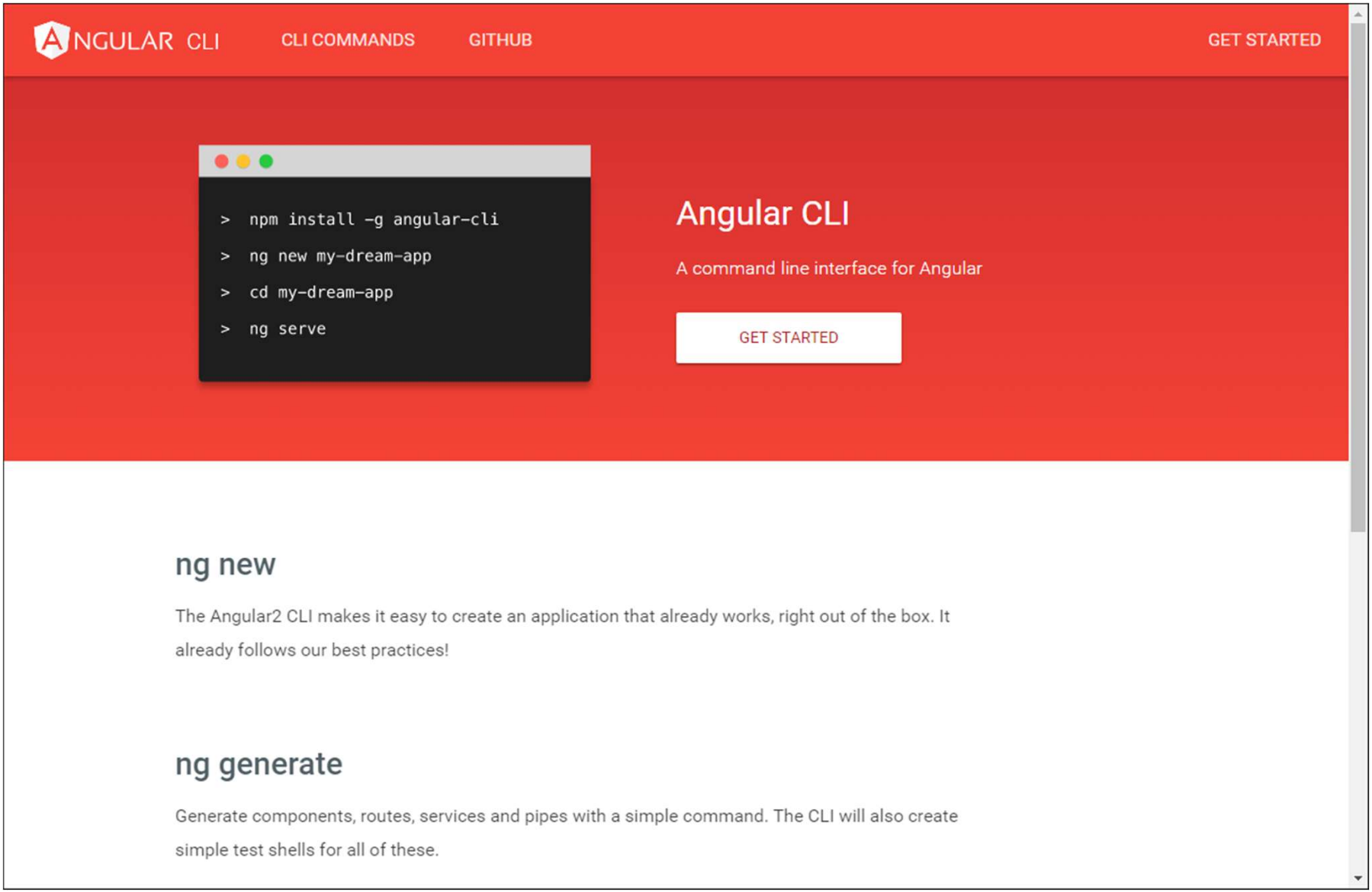
https://github.com/angular/angular-cli

en

https://cli.angular.io/

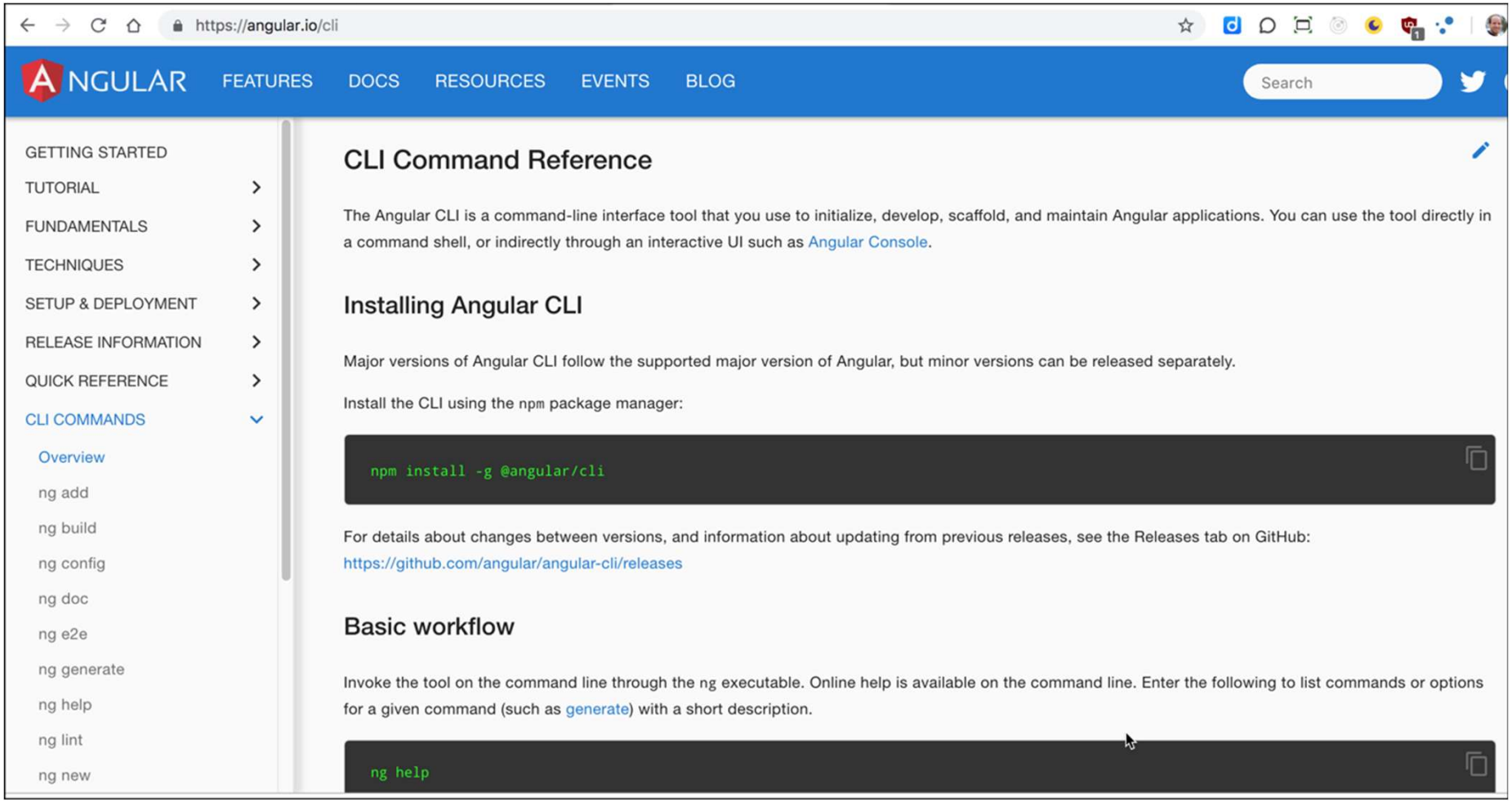



```
npm install -g @angular/cli
```

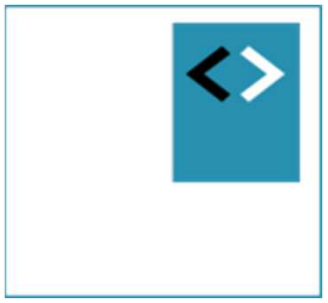

ANGULAR CLI
CLI COMMANDS
GITHUB
GET STARTED
> npm install -g angular-cli
> ng new my-dream-app
> cd my-dream-app
> ng serve
Angular CLI
A command line interface for Angular
GET STARTED
ng new
The Angular2 CLI makes it easy to create an application that already works, right out of the box. It already follows our best practices!
ng generate
Generate components, routes, services and pipes with a simple command. The CLI will also create simple test shells for all of these.

https://www.youtube.com/watch?v=wHZe6gGI5RY

Documentatie - in de Angular Docs



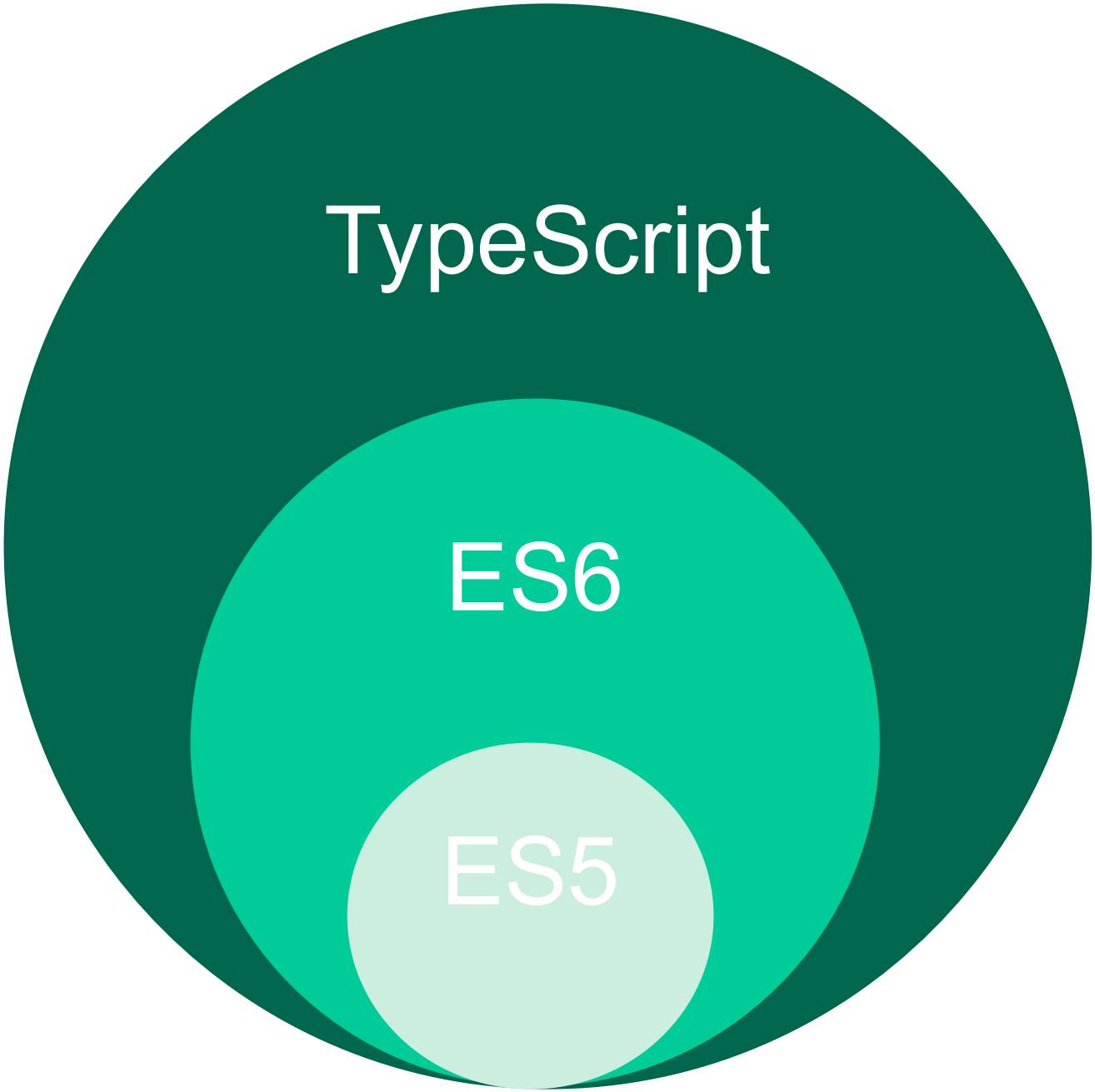
https://angular.io/cli



Angular 2 Code - Backend

Kort over TypeScript en ES6

Programmeertalen


TypeScript
ES6
ES5

ES6 en TypeScript

De toekomst van JavaScript is ES6/ES2015

- Major update van JavaScript als programmeertaal
- Modules, classes en meer
- Helpt bij het ontwikkelen in Angular 2

TypeScript breidt ES6 verder uit

- Annotaties & types
- Interfaces
- Compiler

TypeScript – tooling support

Types, Autocompletion.

Compile-time checking in editors.

Alles is *optioneel*. Je kunt altijd nog gewoon JavaScript gebruiken.

Onderdelen van een Component Class

imports

```
import { Component } from '@angular/core';
import { DataService } from './services/data-service';
```

annotations

```
@Component({
  selector: 'orders',
  directives: [DataService],
  templateUrl: 'orders-component.html',
})
```

class

```
export class OrdersComponent {
          …
}
```

Checkpoint

- Angular 2 is een totaal ander framework dan Angular 1
- Component-based vs. Page-based
- Nieuwe syntaxis
- Nieuwe programmeertalen en andere nieuwe kenmerken
- Concepten komen – grotendeels – overeen
- Veel boilerplate-code nodig voor een Quickstart
- Daarna: niet meer naar omkijken. Concentreren op de componenten